

Problem 1

```
package problem1;

public class MyStringLinkedList {

    public class Node {
        String value;
        Node next;
        Node previous;

        Node(Node previous, String value, Node next) {
            this.previous = previous;
            this.value = value;
            this.next = next;
        }

        public String toString() {
            return value;
        }
    }

    Node header;

    MyStringLinkedList() {
        header = null;
    }

    public Node getFirst() {
        return this.header;
    }

    Node get(int index) {
        int i = 0;
        if (index == 0) {
            return this.header;
        }
        if (this.header != null) {
            if (this.header.next != null) {
                Node temp = this.header.next;
                while (temp != null) {
                    if (i == index) {
                        return temp;
                    }
                    temp = temp.next;
                }
            }
        }
        return null;
    }

    public void addSort(String value) {
```

```

        if (this.header == null) {
            this.header = new Node(null, value, null);
            return;
        }
        if (this.header.next == null) {

            if (this.header.value.compareTo(value) > 0) {

                String old_value = this.header.value;
                this.header = new Node(null, value, null);
                this.header.next = new Node(this.header, old_value,
null);

                return;

            }

            this.header.next = new Node(this.header, value, null);
            return;
        }
        Node temp = this.header.next;
        while (temp != null)
        {
            if (temp.next == null) {
                Node nd = new Node(temp, value, null);
                temp.next = nd;
                return;
            }

            if (temp.value.compareTo(value) > 0) {
                Node new_node = new Node(temp.previous, value, temp);
                temp.previous.next = new_node;
                temp.previous = new_node;
                return;
            }

            temp = temp.next;
        }
    }

    public int size() {
        int result = 0;
        if (this.header == null) {
            return result;
        }
        if (this.header.next != null) {
            Node temp = this.header.next;
            while (temp != null) {
                result = result + 1;
                temp = temp.next;
            }
        }
    }

```

```

        return result + 1;
    }

    public boolean isEmpty() {
        return this.header == null;
    }

    public Node getLast() {
        if (this.header == null) {
            return null;
        }

        if (this.header.next == null) {
            return null;
        }

        Node temp = this.header.next;
        while (temp != null) {
            if (temp.next == null) {
                return temp;
            }
            temp = temp.next;
        }
        return null;
    }

    public boolean contains(String item) {
        if (this.header == null) {
            return false;
        }
        if (this.header.next == null && this.header.previous == null) {
            return false;
        }

        if (this.header.value.equals(item)) {
            return true;
        }
        if (this.header.next != null) {
            Node temp = this.header.next;
            if (temp.value.equals(item)) {
                return true;
            }
            while (temp != null) {
                if (temp.value.equals(item)) {
                    return true;
                }
                temp = temp.next;
            }
        }

        if (this.header.previous != null) {

```

```

        Node temp = this.header.previous;
        if (temp.value.equals(item)) {
            return true;
        }
        while (temp != null) {
            if (temp.value.equals(item)) {
                return true;
            }
            temp = temp.previous;
        }
    }

    return false;
}

public void removeFirst() {
    if (this.header != null) {
        if (this.header.next != null) {
            Node temp = this.header.next;
            temp.previous = null;
            this.header = temp;
        }
    }
}

void add(int index, String value) {
    if (this.size() > index) {
        throw new RuntimeException("cannot reach here");
    }
    int i = 0;
    if (this.header != null) {
        if (this.header.next != null) {
            Node temp = this.header.next;
            while (temp != null) {
                if (i == index) {
                    Node temp1 = new Node(temp.previous,
value, temp.next);
                    temp = temp1;
                }
                temp = temp.next;
            }
        }
    }
}

void removeLast() {

```

```

        Node temp = this.header.next;
        while (temp != null) {
            if (temp.next == null) {
                temp.previous.next = null;
                temp.previous = null;
                temp = null;
                return;
            }

            temp = temp.next;
        }
    }

    public void print() {
        print(header);
    }

    // Write a recursive print method to display the elements in the list.
    void print(Node n) {
        if (n == null) {
            System.out.println("");
            return;
        }
        System.out.println(n);
        if (n.next != null) {
            print(n.next);
        }
        return;
    }

    public String toString() {
        if (this.header == null) {
            return "";
        }
        String str = "";
        Node temp = header;
        while (temp != null) {
            str = str + "-->[" + temp.value.toString() + "];"
            temp = temp.next;
        }
        str = str + "-->[" + "NULL" + "];"
        return str;
    }

    public static void main(String[] args) {
        MyStringLinkedList mySL = new MyStringLinkedList();

        String alphabet = "bacdpqrghijklxstuvwefmnoyz";

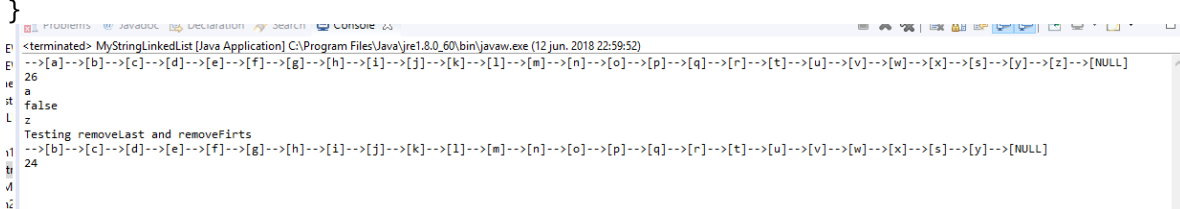
        char[] array = alphabet.toCharArray();
        for (char c : array) {
            mySL.addSort(String.valueOf(c));
        }
    }

```

```

        System.out.println(mySL);
        System.out.println(mySL.size());
        System.out.println(mySL.getFirst());
        System.out.println(mySL.isEmpty());
        System.out.println(mySL.getLast());
        System.out.println("Testing removeLast and removeFirst");
        mySL.removeFirst();
        mySL.removeLast();
        System.out.println(mySL);
        System.out.println(mySL.size());
    }
}

```



Problem 2

```

package problem2;

import java.util.List;
import java.util.ArrayList;
import java.util.Comparator;

class sortMarketingByEmployeeName implements Comparator<Marketing>
{
    public int compare(Marketing a, Marketing b)
    {
        return a.productname.compareTo(b.productname); // Consistency with
        comparator
    }
}

public class Marketing implements Comparator<Marketing> {

    public int compare(Marketing a, Marketing b)
    {
        Double value=a.salesamount-b.salesamount;
        return value.intValue();
    }

    public static List<Marketing> listMoreThan1000( List<Marketing> list)
    {
        List<Marketing> lt = new ArrayList<>();

```

```

        for (Marketing element : list) {

            if(element.salesamount>1000)
            {
                lt.add(element);
            }
        }

        return lt;
    }

    String employeeename, productname;
    double salesamount;
    public Marketing(String employeeename, String productname, double
salesamount) {

        this.employeeename = employeeename;
        this.productname = productname;
        this.salesamount = salesamount;
    }

    @Override
    public String toString() {
        return "Marketing [employeeename=" + employeeename + ", productname="
+ productname + ", salesamount="
        + salesamount + "];"
    }

    @Override
    public int hashCode() {
        final int prime = 741;
        int result = 1;
        result = prime * result + ((this.employeeename == null) ? 0 :
this.employeeename.hashCode());
        result = prime * result + ((this.productname == null) ? 0 :
this.productname.hashCode());
        long temp = Double.doubleToLongBits(salesamount);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
        {
            return true;
        }
        if (obj == null)
        {

```

```

        return false;
    }
    if (getClass() != obj.getClass())
    {
        return false;
    }
    Marketing other = (Marketing) obj;

    return this.salesamount==other.salesamount ;
}

public static void main(String[] args) {
    List< Marketing> database= new ArrayList<>();
    database.add(new Marketing("Juan Fco","Pepsi",800.9));
    database.add(new Marketing("Fernando","Cocacola",200.9));
    database.add(new Marketing("Romario","Sprite",1600.9));
    database.add(new Marketing("Luis","Sprite",5300.9));
    System.out.printf("The database size of marketing is
%s",database.size());
    database.forEach((v)->System.out.println(v.toString()));

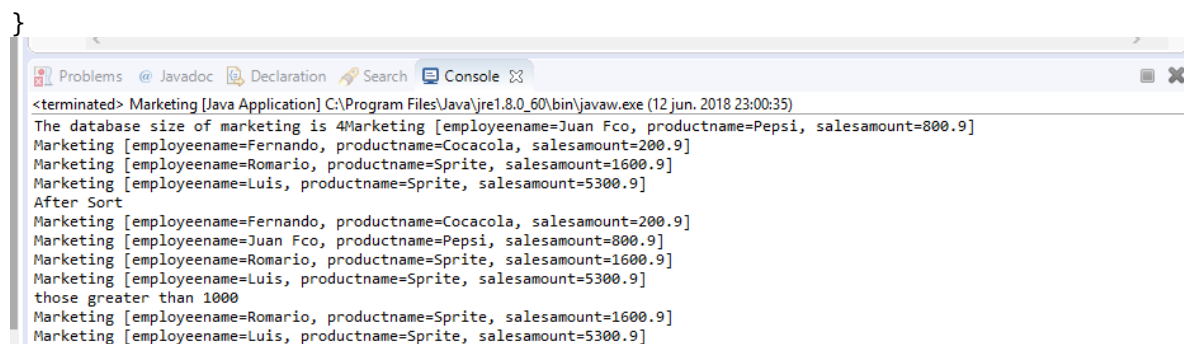
    sortMarketingByEmployeeName comparator= new
sortMarketingByEmployeeName();
    System.out.println("After Sort");

    database.sort(new Marketing(null,null,0));// using interfaces
comparator

    database.forEach((v)->System.out.println(v.toString()));
    System.out.println("those greater than 1000");
    List< Marketing>
database_greater_than_1000=ListMoreThan1000(database);
    database_greater_than_1000.sort(comparator);
    database_greater_than_1000.forEach((v)-
>System.out.println(v.toString()));

}
}

```



```

<terminated> Marketing [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (12 jun. 2018 23:00:35)
The database size of marketing is 4Marketing [employeeename=Juan Fco, productname=Pepsi, salesamount=800.9]
Marketing [employeeename=Fernando, productname=Cocacola, salesamount=200.9]
Marketing [employeeename=Romario, productname=Sprite, salesamount=1600.9]
Marketing [employeeename=Luis, productname=Sprite, salesamount=5300.9]
After Sort
Marketing [employeeename=Fernando, productname=Cocacola, salesamount=200.9]
Marketing [employeeename=Juan Fco, productname=Pepsi, salesamount=800.9]
Marketing [employeeename=Romario, productname=Sprite, salesamount=1600.9]
Marketing [employeeename=Luis, productname=Sprite, salesamount=5300.9]
those greater than 1000
Marketing [employeeename=Romario, productname=Sprite, salesamount=1600.9]
Marketing [employeeename=Luis, productname=Sprite, salesamount=5300.9]

```